

Android'de Kanca Atmak

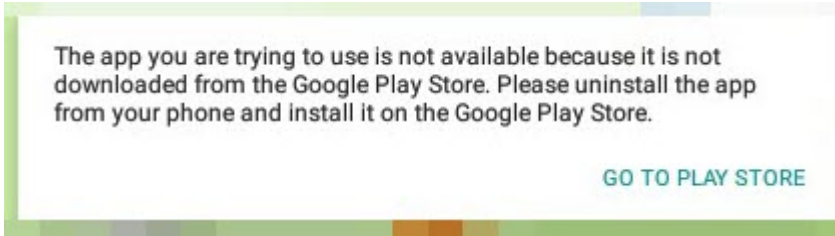
written by Mert SARICA | 1 January 2021

If you are looking for an English version of this article, please visit [here](#).

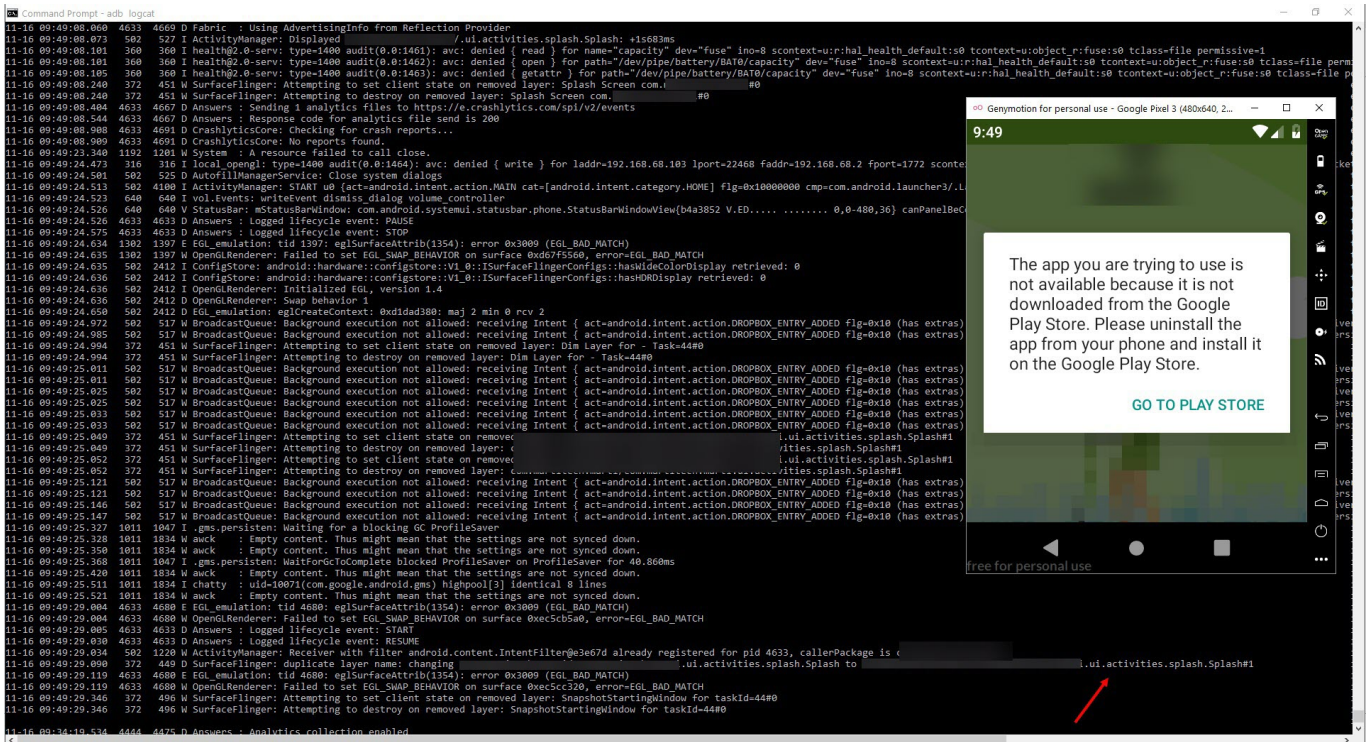
Konumuz Android dünyası olsa da kanca atma denilince nedense aklıma ilk olarak enerji nakil hattına kanca atılarak hanelere çekilen kaçak elektrik gelir. Android dünyasında da aslında uygulamaları dinamik olarak analiz etmek veya müdahale etmek istediğimizde de benzer bir yöntem izleriz. Peki buna neden ihtiyaç duyarız ? Kimi zaman Android uygulamaları ile ilgili bir güvenlik araştırması yapmak istediğimizde veya sızma testi esnasında güvenlik zafiyeti bulmak için hedef Android uygulamasını analiz etme ihtiyacı duyarız. Bunun için genellikle ilk olarak hedef Android uygulamasını kaynak koduna çevirip statik kod analizi yapmakla işe başlarız. Fakat günümüzde çoğu Android uygulamasında kodlar gizlenerek (obfuscation) anlaşılması güçleştirildiği için uygulamayı Genymotion gibi öykünücüler (emulator) üzerinde dinamik olarak analiz etmeye çalışırız. Çalışırız dememin bir sebebi ise yine günümüzde Snapchat gibi mobil uygulamaların statik analizin yanı sıra dinamik analizi de engelleme adına çok sayıda yöntemle başvurduğu görülebilmektedir.

2019 yılının Aralık ayında gerçekleştirdiğim Stockholm seyahatim esnasında şehri gezerken etrafımdan sık şıkırdım insanların vızır vızır elektrikli scooterlar ile geçip gittiğini gördüm. Avrupa'da kullanımının son derece yaygın olması sebebiyle elektrikli scooter uygulamalarının zaman içinde güvenlik araştırmacılarının radarına girmesi ile bazılarında güvenlik zafiyetlerinin tespit edildiğini hatırladım. Ülkemizde de yaygınlaşmaya başlayan elektrikli scooterları ve uygulamalarını kullanmaya başlamadan önce Android uygulamalarından bir tanesinin haberleşmesini, merakımı gidermek amacıyla güvenlik araştırmacısı gözüyle incelemeye karar verdim. Tabii ki her zaman olduğu gibi evdeki hesap çarşıya uymadı ve karşıma çıkan engeller sayesinde ortaya bu blog yazısı çıkmış oldu. :)

Zamanı kısıtlı bir güvenlik araştırmacısı olarak statik kod analizi ile vakit kaybetmek istemediğim için APK dosyasını APK Downloader web uygulaması ile indirip Geny Motion öykünücüsüne yükledim. Uygulamayı çalıştırdıktan sonra karşıma APK dosyasının Google Play üzerinden indirilmemesi sebebiyle sürpriz bir uyarı mesajı çıktı. :)



Bunun üzerine gözü kapalı statik kaynak kodu analizine girmeden önce komut satırında sistem mesajlarını görüntülemek için adb logcat komutunu çalıştırıp ardından da uygulamayı tekrar çalıştırdım. Mesajlar arasından ekrana gelen uyarı mesajı ile ilişkili olduğunu düşündüğüm fonksiyonu bulmak için jadx aracı ile APK dosyasını kaynak koduna çevirdikten sonra activities.splash.Splash dosyasını incelemeye başladım. Dosyanın sonunda yer alan verifyInstallerId fonksiyonu hemen dikkatimi çekti. Bu fonksiyon adını Google arama motorunda arattığımda benzer bir fonksiyonun tam da bu amaçla kullanıldığını gördüm. Bu fonksiyon ile Android'in getInstallerPackageName fonksiyonundan faydalanarak uygulamayı Android'e yükleyen uygulamanın Google Play olup olmadığı kontrol ediliyordu. Şayet uygulama Google Play tarafından işletim sistemine yüklendiyse installerPackagename değişkeni sıfırdan farklı bir değer oluyordu.



The image is a screenshot of a computer screen. The top half shows an IDE window with a Java file named `ui.activities.splash.Splash`. The code is for an Android activity that checks if the app is installed from the Play Store. The code includes methods for getting the content view, context, and initializing the view. It also has a `verifyInstallerId` method that checks if the app is installed from the Play Store by comparing the installer package name with a list of valid installers.

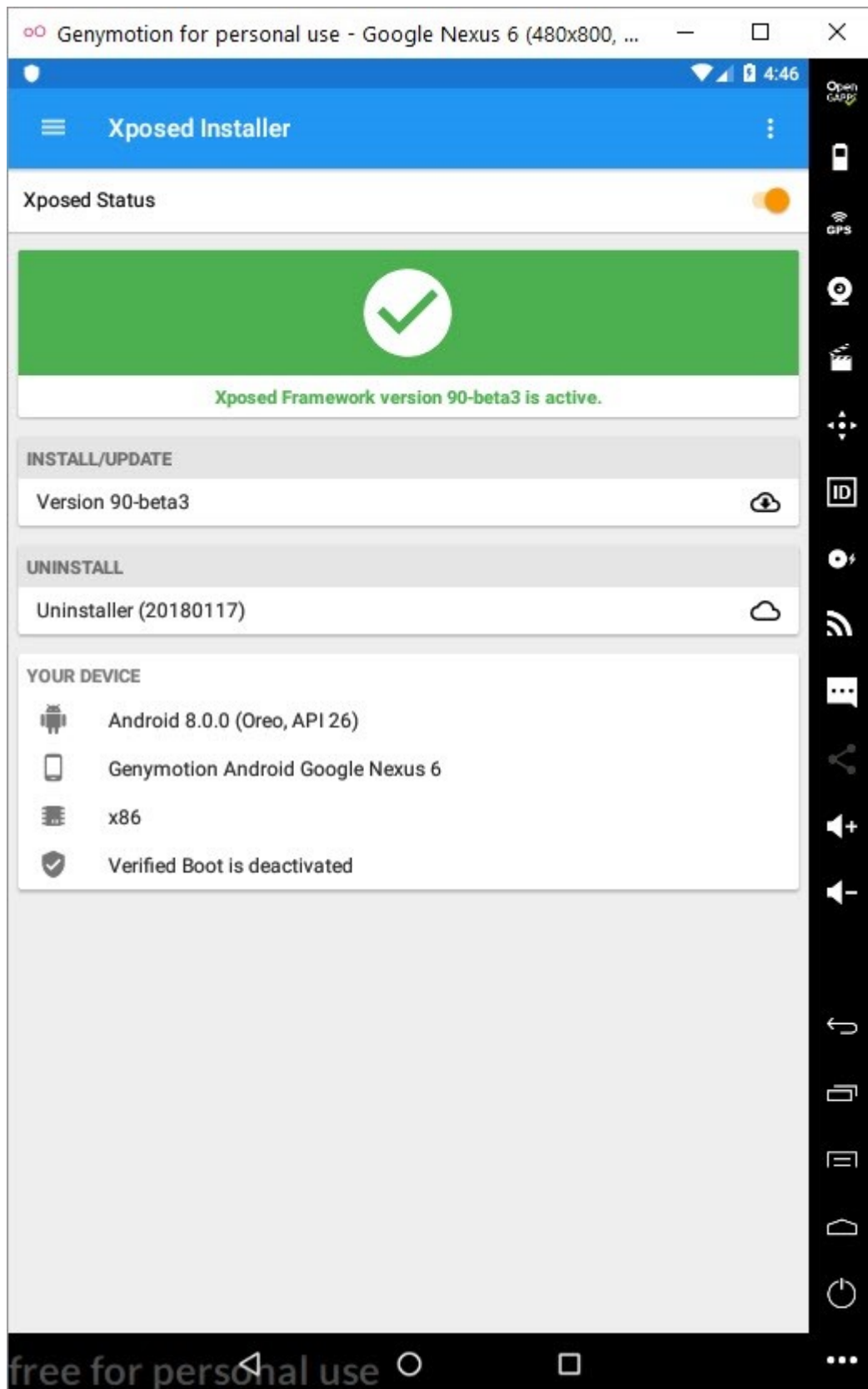
```
124 public int getContentView() {
125     return R.layout.activity_splash;
126 }
127
128 public Context getContext() {
129     return this;
130 }
131
132 public void initView() {
133     if (this.presenter == null) {
134         this.presenter = new SplashPresenter(this);
135     }
136     if (!verifyInstallerId(this)) {
137         showWrongAppVersion();
138         return;
139     }
140     this.presenter.getConfig();
141     setSnackBarView(FindViewById(R.id.rootlayout), true);
142     this.versionCode = 75;
143 }
144
145 public void onError(String str) {
146     if (!"inProgress".equals(str)) {
147         hideProgress();
148         if (!str.equals("") && !str.equals(Constants.EXCEPTION)) {
149             showAlert(str);
150         }
151     } else if (progressIsShown()) {
152         setProgressMessage();
153     }
154 }
155
156 public void onHasRide(boolean z) {
157     if (z) {
158         gotoActiveRide();
159     } else {
160         gotoHomePage();
161     }
162 }
163
164 public void onLoadConfig() {
165     if (LocalDataManager.getInstance().getConfig().getAndroidVersion() > this.versionCode) {
166         showUpdateAlert();
167     } else {
168         checkLogin();
169     }
170 }
171
172 public boolean verifyInstallerId(Context context) {
173     ArrayList<String> validInstallers = new ArrayList<>(Arrays.asList("com.android.vending", "com.google.android.feedback"));
174     String installerPackageName = context.getPackageManager().getInstallerPackageName(context.getPackageName());
175     return installerPackageName != null && validInstallers.contains(installerPackageName);
176 }
177
178 }
```

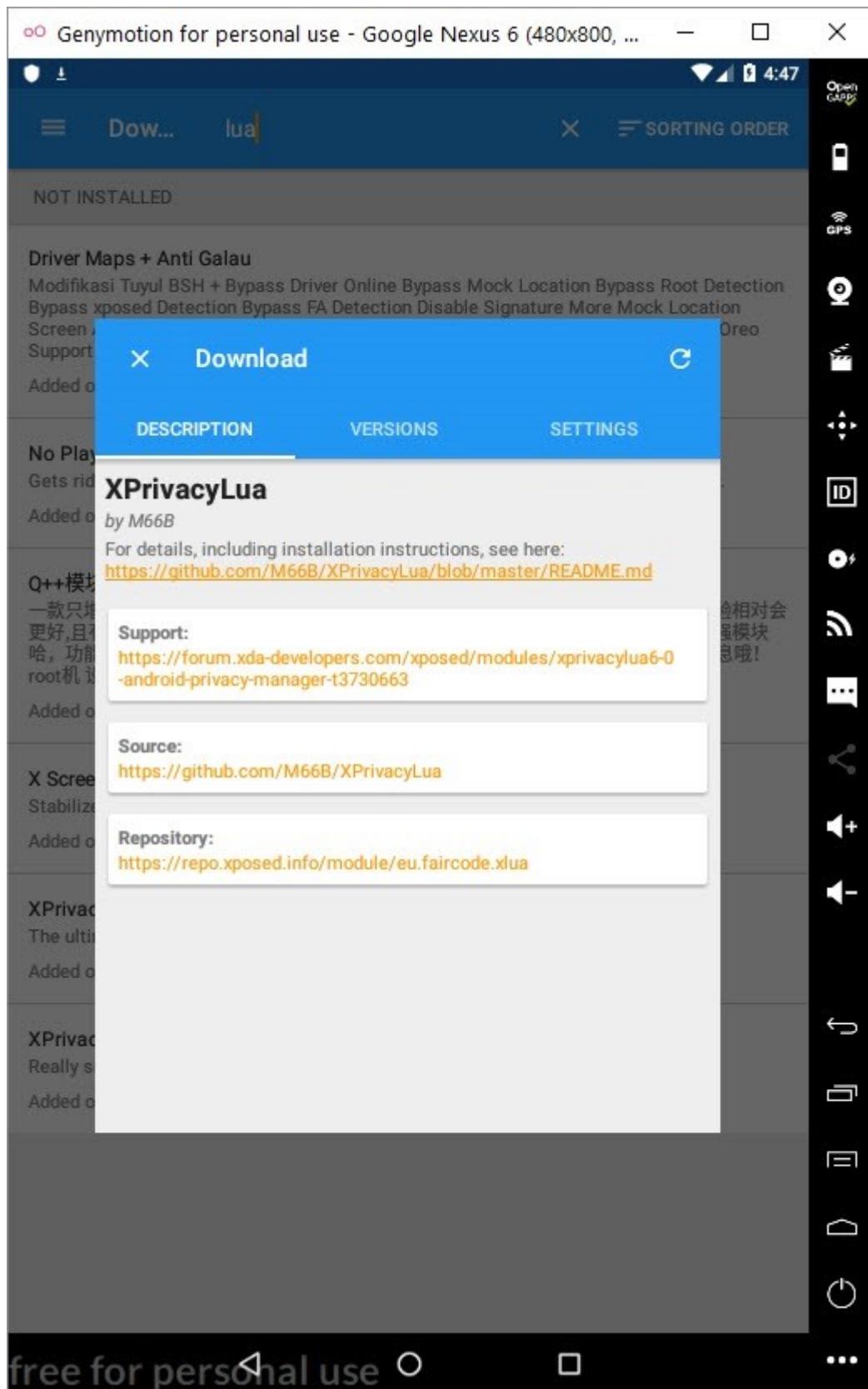
The bottom half of the screen shows a Stack Overflow page titled "Detect if an app is installed from Play store". The question asks how to prevent a user from using an app if it has not been downloaded from the Google Play store. The answer, provided by Javier S., shows the same `verifyInstallerId` method as in the IDE. The page also includes a sidebar with navigation links, a search bar, and a list of featured questions and answers.

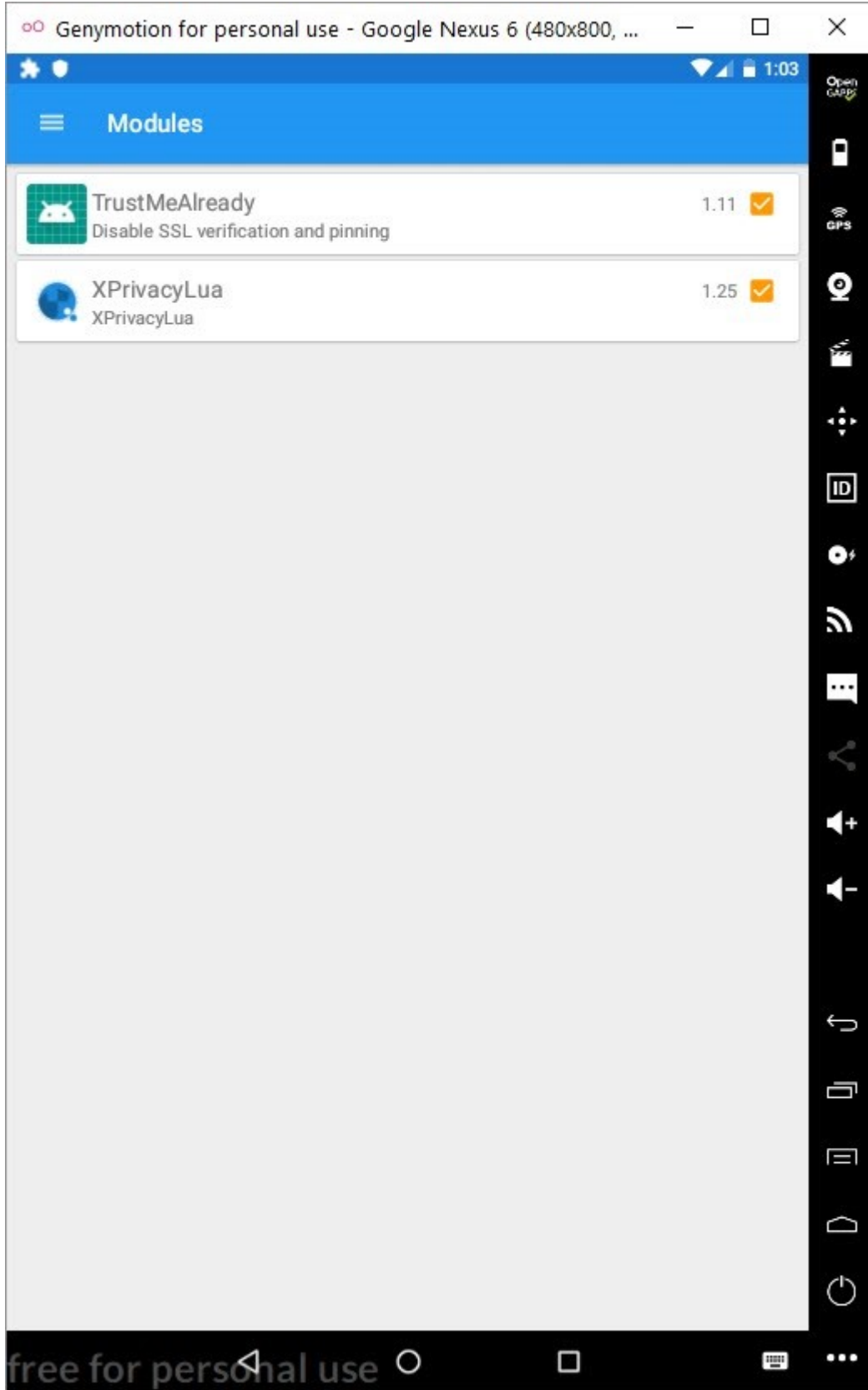
Bu fonksiyona kaynak kodu seviyesinde müdahale edip, `installerPackageName` değişkenini sıfırdan farklı bir değer yapıp ardından derleyip Android işletim sistemi üzerinde çalıştırabilirdim fakat Bill Gates'in bir röportajında dediği gibi "Her zaman en tembel insanları işe alırım çünkü tembeller çok karışık işleri bile en kısa yoldan yaparlar" ben de tembellik yapıp kısa bir yol aramaya karar verdim. :)

Eminim bu gibi bir durumla karşılaşan güvenlik araştırmacılarının çoğu Frida araç kiti ile ilerlemeyi tercih ederler fakat hayatın Frida'dan ibaret olmaması gerektiğine inanarak Frida'ya alternatif bir araç, farklı bir yol aramaya karar verdim. Google arama motorunda kısa bir araştırma yaptıktan sonra daha önce sızma testlerinde özellikle SSL Pinning'i atlatmak için kullandığım ve 1400'den fazla eklentiye sahip olan Xposed Framework aklıma geldi.

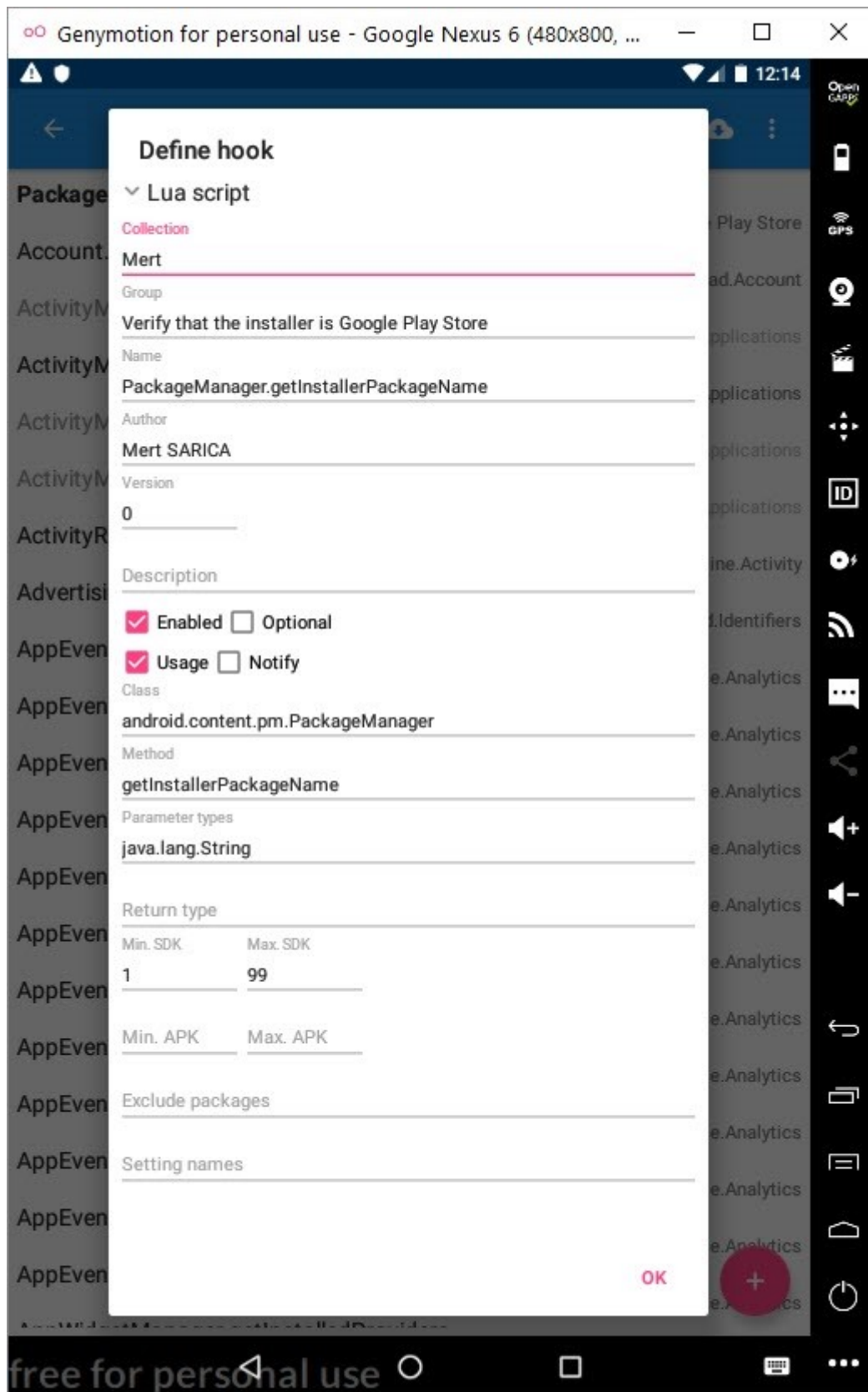
Geny Motion üzerinde bulunan Android Oreo işletim sistemine Xposed Framework'u kurduktan sonra eklentilerine göz atmaya başladım. Eklentiler arasında XPrivaclyLua isimli eklenti hemen dikkatimi çekti. Adından da anlaşılabilirceği üzere bu eklenti, Android üzerinde yüklü olan uygulamaları sahte bilgilerle (sahte konum bilgisi gibi) besleyerek mahremiyetinizi korumaya yardımcı olmaktadır. Çalışma yöntemi olarak kabaca bu bilgileri toplamaya çalışan fonksiyonlara kanca atarak gerçek bilgiler yerine sahte bilgiler vermektedir. Eklentiye ihtiyaçlarınız doğrultusunda şekillendirmek için ise Pro sürümünü yükleyerek Lua programlama dili ile betikler oluşturmanız gerekmektedir.

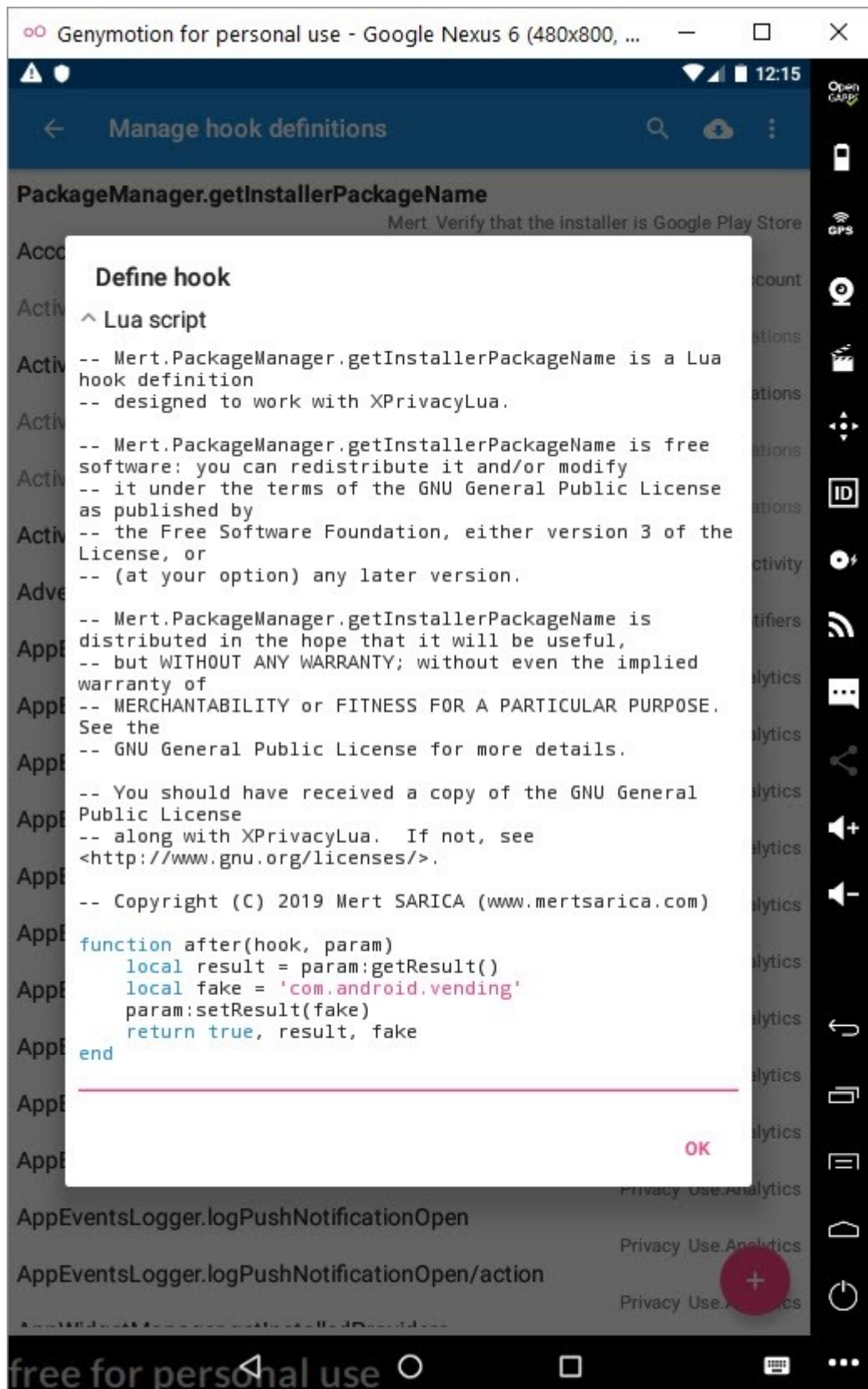


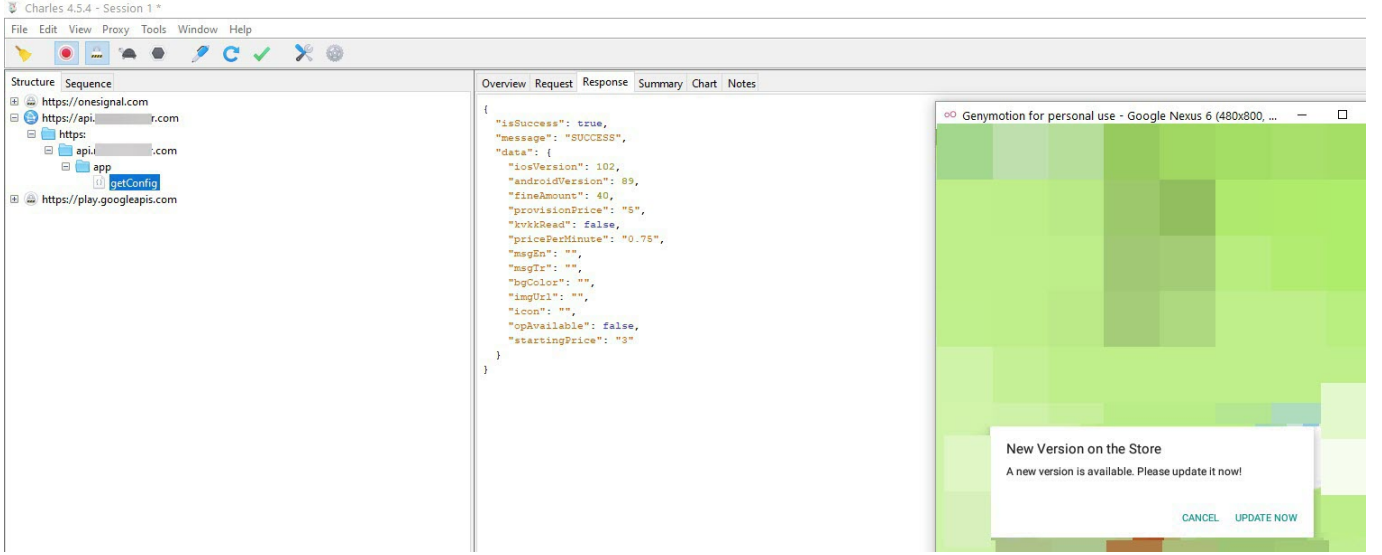




Lua ile `installerPackagename` değişkenini değiştiren ufak bir betik hazırlayıp aktif hale getirdikten sonra uygulamayı çalıştırdığımda uygulamanın artık daha önceki uyarı mesajını çıkarmadığını ve Charles Proxy ile web trafiğini görüntüleyebildiğimi görerek mutlu sona ulaşmış oldum.







Bu yazının güvenlik arařtırmalarında Frida'ya alternatif araç ve yöntem arayanlara ışık tutacağını ümit ederek bir sonraki yazıda görüşmek dileğiyle herkese güvenli günler dilerim.